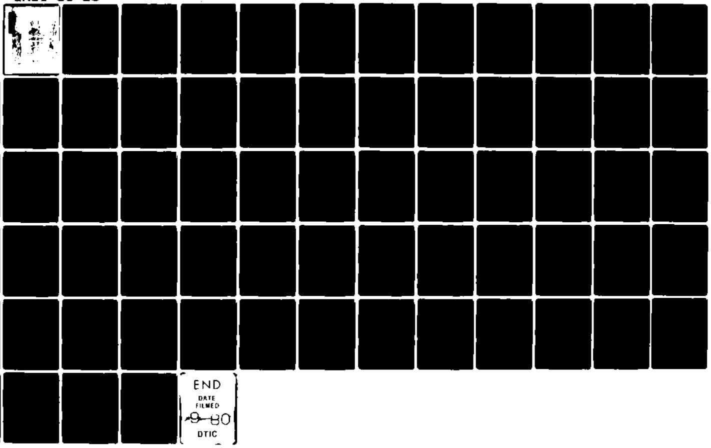
AD-A087 196 WOODS HOLE OCEANOGRAPHIC INSTITUTION MASS F/G 8/1
A DATABASE FOR ZOOPLANKTON NET TOW DATA.(U)
JUN 80 M M HUNT, P H WIEBE N00014-79-C-0071
UNCLASSIFIED WHOI-80-28 NL

1 OF 1
AD A
087196



END
DATE
FILMED
9-80
DTIC

UNCLASSIFIED 6/80

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
~~1. REPORT NUMBER~~ 14 WHOI-80-28	2. GOVT ACCESSION NO. AD-A087 196	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 A DATABASE FOR ZOOPLANKTON NET TOW DATA		~~5. TYPE OF REPORT & PERIOD COVERED~~ 9 Technical Rept.
		~~6. PERFORMING ORG. REPORT NUMBER~~
~~7. AUTHOR(s)~~ 10 Mary M. Hunt [redacted] Peter H. Wiebe		~~8.~~ CONTRACT OR GRANT NUMBER(s) 15 N00014-79-C-0071
9. PERFORMING ORGANIZATION NAME AND ADDRESS Woods Hole Oceanographic Institution Woods Hole, MA 02543		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 083-004
11. CONTROLLING OFFICE NAME AND ADDRESS NORDA National Space Technology Laboratory Bay St. Louis, MS 39529		~~12. REPORT DATE~~ 11 June 1980
		13. NUMBER OF PAGES 65
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 12 73		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

1. MOCNESS Net Tows
2. CODASYL Database for Zooplankton
3. Zooplankton Data Storage and Retrieval

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the design and implementation of a database to store zooplankton net tow data and the applications programming done to update and access the database using the Sigma 7 Extended Database Management System. The database contains information about each tow (cruise name, tow number, type of tow, year, month, day, time of day, longitude, latitude, area of tow, day-night code); each sample (depth code; minimum and maximum depth; minimum, maximum and average values of temperature, salinity, oxygen, (Con. on back

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE UNCLASSIFIED 6/80

S/N 0102-014-6601

391000 SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered) light and chlorophyll; total biomass, aliquot size, and volume of water filtered); each species (family, genus, species names, and catch per 1000 m^3]. Information can be retrieved by user-written applications programs or with the Sigma 7 Interactive Database Processor, which can either print a report of the retrieved data or store it in a file for further processing. As presently formulated, the database can store up to 500 tows or samples, 10 families, 100 genera, 500 species and 50,000 catch records.

cu m.

A

Accession For	
NTIS GRA&I	✓
DDC TAB	
Unannounced	
Justification____	
By____	
Distribution/	
Availability Codes	
Dist	Avail and/or special
A	

WHOI-80-28

A DATABASE FOR ZOOPLANKTON NET TOW DATA

by

Mary M. Hunt

and

Peter H. Wiebe

WOODS HOLE OCEANOGRAPHIC INSTITUTION
Woods Hole, Massachusetts 02543

June 1980

TECHNICAL REPORT

Prepared for the Office of Naval Research under Contract N00014-79-C-0071; NR 083-004.

Approved for Distribution: [illegible]
George D. Grice, Chairman
Department of Biology

A DATABASE FOR ZOOPLANKTON NET TOW DATA

CONTENTS

Abstract

This report describes the design and implementation of a database to store zooplankton net tow data and the applications programming done to update and access the database using the Sigma 7 Extended Database Management System. The database contains information about each tow (cruise name, tow number, type of tow, year, month, day, time of day, longitude, latitude, area of tow, day-night code); each sample (depth code; minimum and maximum depth; minimum, maximum and average values of temperature, salinity, oxygen, light and chlorophyll; total biomass; aliquot size; and volume of water filtered); each species (family, genus, species names, and catch per 1000 m^3). Information can be retrieved by user-written applications programs or with the Sigma 7 Interactive Database Processor, which can either print a report of the retrieved data or store it in a file for further processing. As presently formulated, the database can store up to 500 tows or samples, 10 families, 100 genera, 500 species and 50,000 catch records.

Keywords: MOCNESS Net Tows, CODASYL Database for Zooplankton, Zooplankton Data Storage and Retrieval.

Woods Hole Oceanographic Institution Project Number 10/71.54

I. Introduction

Extensive sampling with nets of the zooplankton populations in the Northwestern Atlantic Ocean between 1972 and 1978, and subsequent laboratory analysis of the samples, have resulted in a large amount of information about the abundance of species in the samples and the physical circumstances under which the samples were collected. In order to analyze these data efficiently, it became obvious that a database and retrieval system had to be constructed. This report documents the implementation of such a database and the construction of applications programs to update and access the database. This database relies on the Extended Database Management System (EDMS) which is a part of the operating system of the Honeywell Sigma 7 Computer at the Woods Hole Oceanographic Institution. The remainder of this introduction is a brief description of the EDMS system.

The Extended Database Management System consists of several different processors, and a group of library subroutines. The steps involved in database creation and use are:

1. The first step is to prepare a formal description of the database to be created, using Data Definition Language (DDL). This description is processed by the File Definition Processor, DMSFDP. Two random files are created: a schema file which is a description of the database used by other processors, and a subschema file which is used by the Data Base Manager.

2. The second step is to initialize the database, using the processor DMSINIT. This processor uses information from the schema file to create a file having the attributes specified in the DDL, but containing no data.

3. Information is stored in the database, and can be retrieved, by applications programs which use a set of library routines referred to collectively as the Data Base Manager (DBM).

4. The Interactive Database Processor, IDP, can be used to retrieve information from the data base. It can either print a report of retrieved data, or store it in a file.

5. Two processors are supplied which facilitate back-up of the database. Saving the database file is done by DMSDUMP, and restoration is done by DMSLOAD.

6. There are three Data Control Blocks (DCB's) through which the system accesses the various files.

F:SCHE	for the schema file	file name is ZOOSCHEM
F:SSCH	for the subschema file	file name is ZOOPLAN
F:DB01	for the database file	file name is ZOOSTOR

Details about the use of the processors are included later in this report. It should be noted that the name of the file containing the database, and the names of the schema and subschema files are contained in the DDL, and cannot be changed.

The following section of this report (section II) contains a complete description of the database. The following sections give details about how to use the database. The entire report, except perhaps section X, should be read carefully before attempting to store information in the database. It cannot be emphasized too strongly that extreme care should be taken in preparing data to be stored in the database.

II. Database Description

Our database contains information about the results of plankton tows. This information includes data describing the tows, and individual samples within the tows. It also includes the number of each species found in each sample.

The reasons for storing this information in a database are:

1. To avoid repetition of genus and species, and tow and sample numbers, in every catch record.
2. To facilitate processing either by sample or by species. Some applications compare the catch from different samples, while others study the samples in which a specific species was found.
3. Relative ease of adding new data.

The database is defined in terms of groups and sets. A graphic representation of the database is given in Figure 1. Each group defines a collection of related attributes which bear a one-to-one relationship to each other. For example, information about tows is a group; actual data about one specific tow is called a group occurrence. In Figure 1, each group is represented by a box; the group-name is printed in large letters in the middle of the box. The first group, ZOOHEAD, is a header group used by the Data Base Management routines, and contains no actual data.

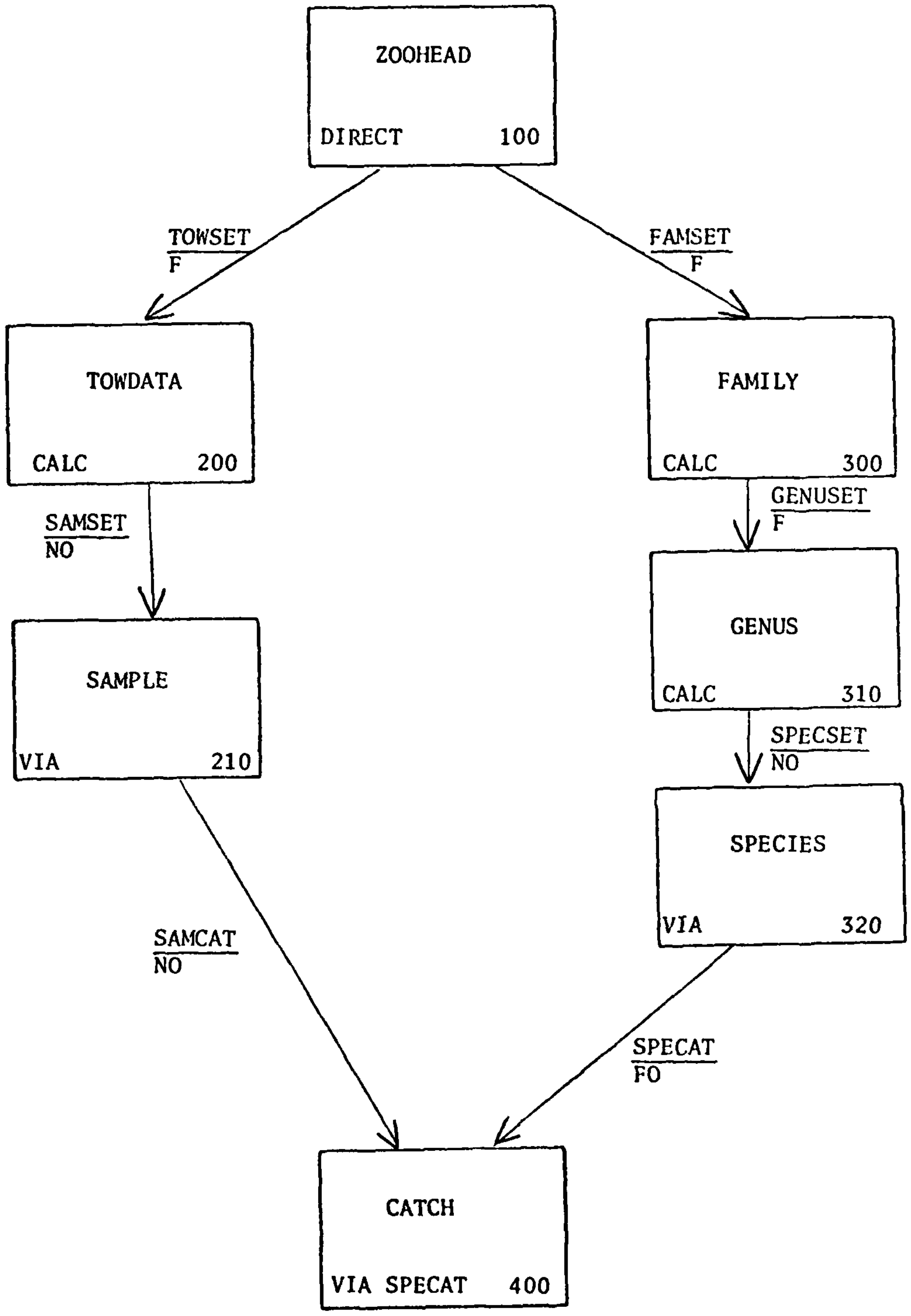
The arrows in Figure 1 show the links between groups. These links are called sets; the set-name is printed beside the arrow, and is underlined. Each set points from one owner group to one member group. This connects one occurrence of the owner group to one or more occurrences of the member group. For example, an occurrence of the TOWDATA group (information about a specific tow), would be linked with one or more occurrences of the sample group (information about samples from the tow). The set linkages allow backward access; it is possible for a given SAMPLE occurrence, to find the owner from the TOWDATA group.

These links make it possible for a user program to access all samples (and tows) in which a given species occurred, or to find all species present in a given sample.

The order in which group occurrences are accessed depends on the database definition and on the order in which the occurrences were stored. Occurrences of the FAMILY, GENUS, and TOWDATA groups will be accessed in the reverse order to which they were stored. Occurrences of these groups can also be accessed individually by name. Occurrences of the SPECIES and SAMPLE groups will be accessed in the same order in which they were stored. Occurrences of these two groups cannot be accessed except through an occurrence of the group that owns the set of which they are members. For example, it is not possible to access a species name (or the catches associated with it), except through the name of the genus to which the species belongs.

Details of the items included in each group, maximum number of occurrences, and other parameters, are included. Knowledge of the database structure, and items included in each group, are required for use of the Interactive Database Processor (IDP).

Figure 1.


ZOOHEAD
DIRECT 100
TOWSET
F
FAMSET
F
TOWDATA
CALC 200
FAMILY
CALC 300
SAMSET
NO
GENUSET
F
GENUS
CALC 310
SAMPLE
VIA 210
SPECSET
NO
SPECIES
VIA 320
SAMCAT
NO
SPECAT
FO
CATCH
VIA SPECAT 400

GROUP NAME: TOWDATA

CONTENTS: Information pertaining to tow

Member of TOWSET

Owner of SAMSET

	Item Name	Contents	Type	# Words	Comments
*	CRUISE	cruise name	Alpha,4	1	4 characters to identify cruise
*	TOWNUM	tow number	Alpha,4	1	4 character tow number
	TOWTYP	type of tow	Alpha,4	1	4 character tow type
	FAMCODE	code to identify families	Integer	1	each bit corresponds to a family
	YEAR	year	Integer	1	2 digit year
	MONTH	month	Integer	1	2 digit month
	DAY	day of month	Integer	1	
	TIME	time of day	Integer	1	24-hour clock format
	LONGTUDE	longitude	Real	1	+ for east, - for west
	LATUDE	latitude	Real	1	+ for north, - for south
	REGION	area of tow	Alpha,4	1	user-selected code
	NITEDAY	day-night code	Alpha,1	1	D for day, N for night

Total 12 words of data

4 words for pointers

Maximum number of occurrences anticipated = 500

* Control items for retrieval of group occurrences

GROUP NAME: SAMPLE

CONTENTS: Information pertaining to sample

Member of SAMSET

Owner of SAMCAT

Item Name	*Contents*	*Type*	*# Words*	*Comments*
DEPCODE	depth code	Alpha,4	1	To identify integrated samples
DEPMIN	minimum depth	Integer	1	Minimum depth of sample
DEPMAX	maximum depth	Integer	1	Maximum depth of sample
TEMIN	minimum temperature	Real	1	
TEAVG	average temperature	Real	1	
TEMAX	maximum temperature	Real	1	
SALTMIN	minimum salinity	Real	1	
SALTAVG	average salinity	Real	1	
SALTMAX	maximum salinity	Real	1	
OXMIN	minimum oxygen	Real	1	
OXAVG	average oxygen	Real	1	
OXMAX	maximum oxygen	Real	1	
LIGHTMIN	minimum light	Real	1	
LIGHTAVG	average light	Real	1	
LIGHTMAX	maximum light	Real	1	
CHLRMIN	minimum chlorophyl	Real	1	
CHLRAVG	average chlorophyl	Real	1	
CHLRMAX	maximum chlorophyl	Real	1	
BIOMASS	total biomass	Real	1	
ALIQUOT	aliquot size	Real	1	Fraction of sample studied (8=1/8)
VOLFIL	volume of water	Real	1	In cubic meters
UNDEF	undefined	Integer	5	Room for expansion
		Total	26	words of data
			4	words for pointers

Maximum number of occurrences anticipated = 500

GROUP NAME: FAMILY

CONTENTS: Names of Families included in data base

Member of FAMSET

Owner of GENUSET

Item Name	Contents	Type	# Words	Comments
* FAMNAME	Family name	Alpha,17	5	Names can have up to 17 characters
		Total	5	words of data
			4	words for pointers

Maximum number of occurrences anticipated = 10

* Control item for retrieval of group occurrence.

GROUP NAME: GENUS

CONTENTS: name of each Genus included in database

Member of GENUSET

Owner of SPECSET

Item Name	Contents	Type	# Words	Comments
* GENAME	Genus name	Alpha,18	5	Up to 18 characters

Total 5 words of data

4 words for pointers

Maximum number of occurrences anticipated = 100

* Control item for retrieval of group occurrence.

GROUP NAME: SPECIES

CONTENTS: name of each Species found

Member of SPECSET

Owner of SPECAT

Item Name	Contents	Type	# Words	Comments
SPENAME	species name	Alpha,16	4	Up to 16 characters for species

Total 4 words for data

4 words for pointers

Maximum number of occurrences anticipated = 500

GROUP NAME: CATCH

CONTENTS: number of species in sample

Member of SPECAT

Member of SAMCAT

Item Name	Contents	Type	# Words	Comments
NPCUM	catch per 1000 cubic meters	Real	1	Can always go back to count

Note: For integrated samples (Depth code = 99) this means catch per 1000 meters of depth per 1000 cubic meters.

Total	1	word of data
	4	words for pointers

Maximum number of occurrences anticipated = 50,000

III. Recommended Operating Procedures

A database is different from an ordinary file. It is not possible to enter, delete, or change information, except through software created for the purpose. Program ZOOPUP has been written to update the ZOOSTOR database. Before using ZOOPUP to add information, Program SPLECK should be used to check for errors in the input file.

The operations required fall into two categories: those that must be done once to get started, and those that are repeated for every update. Operations in the first category are:

1. Run DMSFDP to create the schema and subschema files.
2. Run DMSINIT to create and initialize the database file.
3. Run ZOOPUP to add Family, Genus, and Species names to the database.

Operations which will be performed periodically are:

1. Run Program SPLECK to check the input file.
2. Run Program PLANKINT if necessary, to create integrated samples.
3. Run ZOOPUP to add new information to the database.
4. Use IDP, or a test program, to see if the previous operation was successful.
5. Use DMSDUMP to save database on labelled tape.

Accessing the information can, of course, be done at any time, and presents no danger to the integrity of the data.

The following recommendations should be given consideration:

1. Keep all runs, plainly labelled as to what they are.
2. Keep a notebook of updates.
3. Do not enter data directly from cards. Always copy the cards to a file, and use Program SPLECK to check for errors, until none are found.

4. Do not try to add all your data in one run. Try one cruise at a time.

5. Be sure to use FILECAT as needed, so your schema, subschema, and database files will not be purged.

6. Keep two labelled tapes for backup with DMSDUMP. Alternate their use, so you will always have the two most recent updates.

IV. To Initialize, Store, and Restore Database Use of Auxiliary Processors

This section describes the use of processors to initialize, save, and restore a database.

A. DMSFDP

The first step is to process the Data Definition Language description of the database, and create the schema and subschema files. This is accomplished by DMSFDP. The following job set-up should be used:

```
!JOB
!LIMIT (CORE,20)
!SET M:SI /FILEDESC
!DMSFDP
```

The file definition, written in DDL, is read through the M:SI DCB. The schema and subschema files are created. If schema and/or subschema files with the same names already exist, they must be deleted.

B. DMSINIT

This processor must be run before information can be stored in the database. It creates a file having the characteristics specified in the file definition, but containing no data. The job set-up is:

```
!JOB
!LIMIT (CORE,20)
!SET F:SCHE /ZOOSCHEM
!SET F:DB01 /ZOOSTOR
!DMSINIT
```

DMSINIT can also be used to reinitialize all, or part of, an existing database. In this case, one data card must be added to the above job:

```
AREA=ZOOSTOR  RANGE=(p1,p2).   where p1 and p2 are, respectively,
                               the first and last pages to be
                               reinitialized.
!EOD
```

DON'T forget the period at the end of the data card.

C. DMSDUMP

After every successful update to the database, the new version should be stored on a labelled tape. This is accomplished by DMSDUMP:

```
!JOB
!LIMIT (9T,1),(CORE,20)
!MESSAGE 9T tapid  *WRITE*
!SET F:DUMP LT#tapid/filid
!SET F:SCHE /ZOOSCHEM
!SET F:DB01 /ZOOSTOR
!DMSDUMP
 DUMP.
```

D. DMSLOAD

If a job to update the database aborts for any reason, the database will not be properly closed, and will not be able to be used. When (or if) this happens, it will be necessary to restore the previous version. This is accomplished by DMSLOAD:

```
!JOB
!LIMIT (9T,1),(CORE,20)
!MESSAGE 9T tapid  INPUT
!SET F:LOAD LT#tapid/filid
!SET F:SCHE /ZOOSCHEM
!SET F:DB01 /ZOOSTOR
!DMSLOAD
 LOAD.
```

Note: DMSLOAD will not work correctly if the labelled tape file name is the same as the database file name. Do not use ZOOSTOR as the labelled tape file name.

V. Formats for Input Data

There are two different kinds of files used by the three programs (SPLECK, PLANKINT, and ZOOPUP). The first file contains Family, Genus, and Species names, and the second file contains Tow, Sample, and Catch information. Programs SPLECK and ZOOPUP use both files; Program PLANKINT requires only the tow file.

The family file is very simple. It contains one record for each species. The records should be sorted by Family and Genus. The record format is:

columns	Contents
1-17	Family name
21-38	Genus name
41-56	Species name

The tow file contains four different kinds of records, distinguished by the first four characters of the record. The kinds of records, and record identifiers, are:

T	Tow record
S1	Required (first) sample record
S2	Optional (second) sample record
C	Catch record

These records should be in the following order:

```
Tow record for first tow
     Sample record 1 for first sample of tow
     Sample record 2 for first sample of tow
         Catch record for sample
         Catch record for sample
           :
     Sample record 1 for second sample of tow
     Sample record 2 for second sample of tow
         Catch record for sample
         Catch record for sample
           :
           :
```

```
Tow record for second tow
     Sample record 1 for first sample of tow
     Sample record 2 for first sample of tow
         Catch record for sample
         Catch record for sample

         etc.
```

The formats of the different kinds of records are given in Tables 1, 2, 3, and 4. Figure 2 contains a graphical layout of all four kinds of records.

Samples of both kinds of files are included.

TABLE 1

Tow Record Format

Columns	Contents
1	T to identify tow data
2-4	blank
5-8	Cruise name, 4 characters alpha
9	blank
10-13	Tow name, 4 characters alpha
14	blank
15-18	Tow type, 4 characters alpha
19	blank
20-21	Year, 2 digits
22-23	Month, 2 digits
24-25	Day of month, 2 digits
26	blank
27-30	Time, 24-hour clock
31	blank
32-35	Region, 4 characters alpha
36	Day-night code, 1 character alpha
37	blank
38-44	Longitude, of form ±xxx.xx
45	blank
46-51	Latitude, of form ±xx.xx
52	blank
53-72	In each 2 columns, punch a code corresponding to one of the Families identified. If only one Family, there will be only 1 code. Could be: 01 = Euphausids 02 = Copepods

TABLE 2

Format of Sample Record 1

Columns	Contents	
1-2	S1 to identify first sample record	
3-4	blank	
5-8	Depth code, up to 4 characters, alpha	
9	blank	
10-14	Minimum depth (right-justify)	
15	blank	
16-20	Maximum depth (right-justify)	
21	blank	
22-24	Aliquot size, integer	
25	blank	
26-31	Volume of water filtered (cubic meters)	
32-38	Biomass	
39	blank	
40-44	Minimum temperature	
45	blank	Punch decimal
46-50	Average temperature	xx.xx
51	blank	or
52-56	Maximum temperature	-x.xx

TABLE 3

Format of Sample Record 2

Columns	Contents
1-2	S2 to identify sample record 2
3-4	blank
5-10 11-16 17-22	Minimum, Average, and Maximum salinity form xx.xxx or xx.xx punch decimal
23	blank
24-27 28-31 32-35	Minimum, Average, and Maximum oxygen form x.xx or xx.x punch decimal
36	blank
37-43 44-50 51-57	Minimum, Average, and Maximum light form .xxExx or xx.Exx
58	blank
59-63 64-68 69-73	Minimum, Average, and Maximum chlorophyl form xx.xx or x.xxx punch decimal

---

TABLE 4

Catch Record Format

Columns	Contents
1	C to identify catch record
2-4	blank
5-25	Genus name
26-41	Species name
45-50	Number (free field)

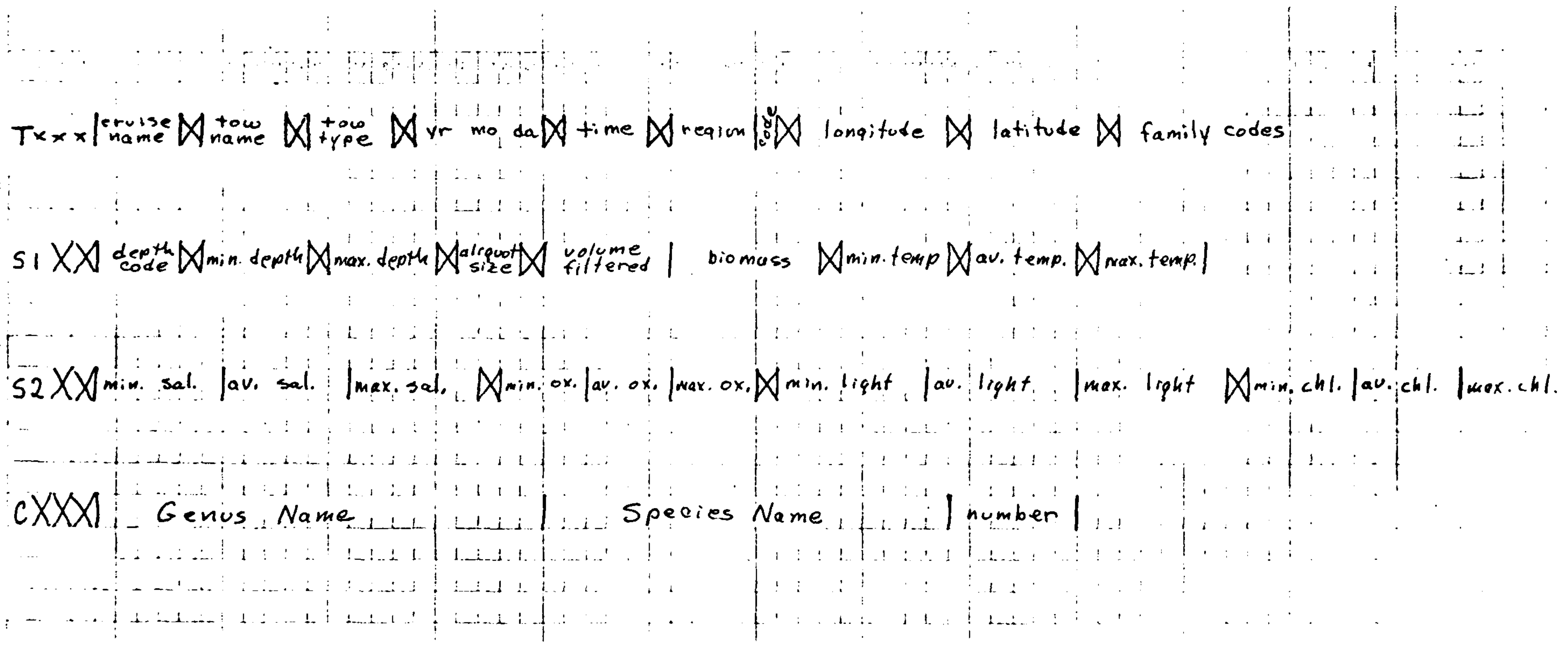

Txxx | cruise name ⊠ tow name ⊠ tow type ⊠ yr mo da ⊠ time ⊠ region | code ⊠ longitude ⊠ latitude ⊠ family codes
S1 XX depth code ⊠ min. depth ⊠ max. depth ⊠ aliquot size ⊠ volume filtered | biomass ⊠ min. temp ⊠ av. temp. ⊠ max. temp. |
S2 XX min. sal. | av. sal. | max. sal. ⊠ min. ox. | av. ox. | max. ox. ⊠ min. light | av. light | max. light ⊠ min. chl. | av. chl. | max. chl.
CXXX Genus Name | Species Name | number |

FIGURE 2.
INPUT FORMATS FOR READTOW

SAMPLE OF: FAMILY/GENUS/SPECIES INPUT

```
EUPHAUSIDS          BENTHEUPHAUSIA          AMBLYOPS
EUPHAUSIDS          EUPHAUSIA               AMERICANA
EUPHAUSIDS          EUPHAUSIA               BREVIS
EUPHAUSIDS          EUPHAUSIA               GIBBOIDES
EUPHAUSIDS          EUPHAUSIA               HEMIGIBBA
EUPHAUSIDS          EUPHAUSIA               KROHNII
EUPHAUSIDS          EUPHAUSIA               MUTICA
EUPHAUSIDS          EUPHAUSIA               PSEUDOGIBBA
EUPHAUSIDS          EUPHAUSIA               TENERA
EUPHAUSIDS          MEGANYCTIPHANES         NORVEGICA
EUPHAUSIDS          NEMATOBRANCHION         BOOPIS
EUPHAUSIDS          NEMATOBRANCHION         FLEXIPIES
EUPHAUSIDS          NEMATOBRANCHION         SEXSPINOSUS
EUPHAUSIDS          NEMATOSCELIS            ATLANTICA
EUPHAUSIDS          NEMATOSCELIS            MEGALOPS
EUPHAUSIDS          NEMATOSCELIS            MICROPS
EUPHAUSIDS          NEMATOSCELIS            TENELLA
EUPHAUSIDS          STYLOCHEIRON            ABBREVIATUM
EUPHAUSIDS          STYLOCHEIRON            AFFINE
EUPHAUSIDS          STYLOCHEIRON            CARINATUM
EUPHAUSIDS          STYLOCHEIRON            ELONGATUM
EUPHAUSIDS          STYLOCHEIRON            LONGICORNE
EUPHAUSIDS          STYLOCHEIRON            MAXIMUM
EUPHAUSIDS          STYLOCHEIRON            SUHMII
EUPHAUSIDS          THYSANOESSA             GREGARIA
EUPHAUSIDS          THYSANOESSA             LONGICAUDATA
EUPHAUSIDS          THYSANOESSA             PARVA
EUPHAUSIDS          THYSANOPODA             ACUTIFRONS
EUPHAUSIDS          THYSANOPODA             AEQUALIS
EUPHAYSIDS          THYSANOPODA             CRISTATA
EUPHAUSIDS          THYSANOPODA             MONOCANTHA
EUPHAUSIDS          THYSANOPODA             OBTUSIFRONS
EUPHAUSIDS          THYSANOPODA             ORIENTALIS
EUPHAUSIDS          THYSANOPODA             PECTINATA
EUPHAUSIDS          THYSANOPODA             TRICUSPIDATA
COPEPODS            PAREUCHEATA             NORVEGICA
```

SAMPLE OF: TOW/SAMPLE/CATCH INPUT

```
T    K062    45 MØC1 761204 2243  CCRN .065.33 +36.12 01
S1   1        900   1000   1     294    85.0   5.00   5.10   5.25
S2   35.00 35.00 35.00
C    THYSANØESSA                LØNGICAUDATA          2
C    THYSANØESSA                PARVA                 2
S1   2        700   900    1     677    26.6   5.25   6.25   7.75
S2   35.00 35.04 35.08
C    NEMATØSCELIS               MEGALØPS              27
C    NEMATØSCELIS               MICRØPS               1
C    THYSANØESSA                PARVA                 10
C    THYSANØPØDA                ACUTIFRØNS            1
S1   3        550   700    1     542    27.7   7.75   8.50  10.00
S2   35.08 35.36 35.38
C    EUPHAUSIA                  TENERA                1
C    NEMATØSCELIS               MEGALØPS              173
C    THYSANØESSA                GREGARIA              1
C    THYSANØESSA                PARVA                 17
S1   4        400   550    1     623    32.1  10.00  10.50  13.25
S2   35.38 35.44 35.68
C    NEMATØSCELIS               MEGALØPS              157
C    NEMATØSCELIS               MICRØPS               1
C    NEMATØSCELIS               TENELLA               1
C    THYSANØESSA                PARVA                 3
S1   5        300   400    1     472    25.4  13.25  14.00  14.75
S2   35.68 35.78 35.86
C    NEMATØSCELIS               MEGALØPS              78
C    STYLØCHEIRØN               ELØNGATUM             1
C    THYSANØESSA                PARVA                 1
C    THYSANØPØDA                ØBTUSIFRØNS           1
S1   6        200   300    1     455    48.4  14.75  15.75  16.75
S2   35.86 36.02 36.17
C    EUPHAUSIA                  KRØHNII               1
C    NEMATØSCELIS               MEGALØPS              60
C    STYLØCHEIRØN               ABBREVIATUM           2
C    STYLØCHEIRØN               AFFINE                5
C    STYLØCHEIRØN               ELØNGATUM             11
C    THYSANØESSA                GREGARIA              2
C    THYSANØPØDA                AEQUALIS              3
S1   7        100   200    1     496    30.2  16.75  17.75  20.30
S2   36.17 36.18 36.36
C    EUPHAUSIA                  HEMIGIBBA             13
C    EUPHAUSIA                  KRØHNII               17
C    EUPHAUSIA                  TENERA                2
C    NEMATØSCELIS               MEGALØPS              35
C    STYLØCHEIRØN               ABBREVIATUM           4
C    STYLØCHEIRØN               AFFINE                10
C    STYLØCHEIRØN               CARINATUM             44
C    STYLØCHEIRØN               ELØNGATUM             2
C    STYLØCHEIRØN               SUHMII                4
C    THYSANØESSA                GREGARIA              1
C    THYSANØESSA                LØNGICAUDATA          1
C    THYSANØPØDA                AEQUALIS              3
S1   8          1   100    1     473    59.2  20.30  20.30  20.30
S2   36.22 36.22 36.22
C    EUPHAUSIA                  AMERICANA             1
C    EUPHAUSIA                  BREVIS                16
C    EUPHAUSIA                  HEMIGIBBA             17
C    EUPHAUSIA                  KRØHNII               12
```

VI. Pre-processing Programs

Two pre-processing programs have been written, one or both of which should be run before additions of data to the database. The first program, SPLECK, should be run at least once before each run of ZOOPUP, to check for errors in the input file. It checks for spelling errors in Genus and Species names, and for valid values in numeric fields. It should be run until no errors are found.

The second program, PLANKINT, is used for tows which include samples from different depths. For each such tow, it creates an additional sample, with a depth code of '99', combining the catch data from all other samples of the tow. For the added sample, the number included on the catch record is the number of species caught per 1000 meters of depth, per 1000 cubic meters of water filtered.

Reports of these two programs follow.

NAME: SPLECK

TYPE: Main program

PURPOSE: This program checks for errors in a file containing tow, sample, and catch information before the data in the file is added to the ZOOSTOR database.

MACHINE: Sigma 7

SOURCE LANGUAGE: Extended Fortran IV

PROGRAM CATEGORY: Data processor

DESCRIPTION:

When storing information in a database, it is extremely inconvenient to encounter errors and inconsistencies in the data to be stored. This program reads information from the files containing input data, and checks for such errors. Only when a file has been found free of errors should it be input to Program ZOOPUP, which will store the information in the database.

There are two files which are required by SPLECK. The first is the file containing Family, Genus, and Species names, called the Family-file. The second file, called the Tow-file, contains information about individual tows and samples, including genus and species names of plankton collected. Each genus-species name must be identical to a genus-species from the Family-file.

Numeric values and other fields which are checked for the different kinds of records in the Tow-files are:

1. Tow Record

   a. Year must be between 60 and 85.
   b. Month, day, time, longitude, and latitude must contain appropriate values.

2. First sample record

   a. Maximum depth > minimum depth
   b. Aliquot > 0
   c. Volume of water > 0
   d. Biomass $\geq$ 0
   e. Min. temp. $\leq$ average temp. $\leq$ max. temp.

3. Second sample record

   For each of the four variables, the following check is made:

   $0 \leq$ minimum $\leq$ average $\leq$ maximum

4. Catch record

   a. The number caught must be > 0.
   b. The genus-species must exist in the Family-file.

In addition, each record is checked to be sure blanks are in the correct locations, and that the record order is legal.

INPUT: A. Up to 200 records of a Family-file are read into the program through the F:1 Data Control Block. The record format is described in Section V of this report.

B. Records from a Tow-file are read through the F:2 Data Control Block. Four kinds of data records are distinguished by T, S1, S2, or C beginning in column 1. The record formats are completely described in Section V of this report.

OUTPUT: Status messages and error messages are written to the printer through the F:108 DCB. These are described under ERRORS & DIAGNOSTICS, below.

USAGE: A. The following job set-up should be used to form a load module.

```
!JOB
!SET F:1 /FAMFIL
!SET F:2 /TOWFIL
!FORTRAN NS,GO
     source deck of SPLECK
!LYNX $;.1JFL;.3 OVER SPLECK
```

B. The job set-up to run the program as a batch job should be:

```
!JOB
!SET F:1 /famfil
!SET F:2 /towfil
!RUN (LMN,SPLECK)
```

C. The program can also be run on-line, by use of the following commands:

```
!SET F:1 /famfil
!SET F:2 /towfil
!S SPLECK
```

RESTRICTIONS:

The Family-file may contain no more than 200 records.

STORAGE REQUIREMENTS:

The program will run with a core limit of 7K.

SUBPROGRAM REQUIRED:

A. Fortran Library: ABORTSET BUFFERIN

B. Library in account 3: DATE SETBREAK COMPAR SCAN

C. Library in account 1JFL (soon to be added to account 3 library):
IDEVICE

OPERATIONAL ENVIRONMENT:

Uses the CP-V Operating System.

Device	Function	Special requirements
disk file	Family-file input	F:1 DCB
disk file	Tow-file input	F:2 DCB
line printer	diagnostic output	F:108 DCB

PROGRAM LOGIC:

A. Initialization

1. Print program name and version
2. Check input files
3. Initialize SETBREAK and ABORTSET
4. Read genus-species names from Family-file and store

B. Process Tow-file

1. Read record and check record type
2. Check record for errors

C. Termination

1. Print summary
2. STOP

TIMING: Undetermined, but fast.

ERRORS & DIAGNOSTICS:

A. The following errors will result in termination of the program.

1. **FAMILY FILE NOT AVAILABLE
SET F:1 WAS NOT ISSUED

2. **TOW FILE NOT AVAILABLE
SET F:2 WAS NOT ISSUED

3. **READ ERROR ON RECORD ____.

This indicates an error in reading the tow file.

4. **FAMILY FILE TOO LONG

There may be no more than 200 records in the family file.

B. For each record from the tow-file which contains one or more errors, three or more lines are printed. The first line contains the digits 1234567890 repeated across the page, to identify column numbers. The second line contains the record number, and contents of the record. The third line (and additional lines if needed) tells the kind of error detected. These messages are:

1. RECORD OUT OF ORDER
2. UNRECOGNIZED RECORD TYPE
3. BLANK CHARACTER NOT FOUND AT COLUMN ___
4. GENUS/SPECIES NAME NOT IN FAMILY FILE
5. A variable name and value, if the variable has an incorrect value.

C. If a DECODE error occurs, the system will print an error message. This is followed by a program message:

ERROR OCCURRED ON RECORD #__________

Sample output, including error messages of different kinds, is included.

PROGRAMMER: John F. Loud and Mary Hunt

ORIGINATOR: Peter Wiebe

DATE: November, 1979

REFERENCES: Complete documentation of ZOOSTOR database.
Reports of W.H.O.I. Sigma 7 programs.

SAMPLE OUTPUT FROM SPLECK:

```
SPLECK  VERSION 1:        12/03/79  13:37:47

NUMBER OF FAMILY RECORDS READ:    35

          123456789012345678901234567890123456789012345678901234567890123456789012
#  29  C    NEMATOBRACHION          BOOPIS                   1
 **GENUS/SPECIES NAME NOT IN FAMILY FILE

          123456789012345678901234567890123456789012345678901234567890123456789012
#  57  T    A271    7 MTRN  720924 2245  NSSN -68.28  35.16
**BLANK CHARACTER NOT FOUND AT COLUMN #14

          123456789012345678901234567890123456789012345678901234567890123456789012
# 123  T    A271    18 MTRN 720927 0445  NSSN -68.27  96.23
LATITUDE = 96.2300

          123456789012345678901234567890123456789012345678901234567890123456789012
# 134  D   STYLOCHEIRON          CARINATUM               25
**UNRECOGNIZED RECORD TYPE

          123456789012345678901234567890123456789012345678901234567890123456789012
# 157  S1     99  1000   805   2   2195
MINDEPTH = 1000
MAXDEPTH = 805

          123456789012345678901234567890123456789012345678901234567890123456789012
# 246  C   EUPHAUSIA             BREVIS                   3
**RECORD OUT OF ORDER

          123456789012345678901234567890123456789012345678901234567890123456789012
# 279  C   MEGANYTIPHANES        NORVEGICA                1
 **GENUS/SPECIES NAME NOT IN FAMILY FILE.

**END OF FILE ON DATA RECORD INPUT AFTER   279 RECORDS PROCESSED
**       7 ERRORS DETECTED

*STOP* 0
```

NAME: PLANKINT

TYPE: Main program

PURPOSE: To find integrated plankton tows from all samples.

MACHINE: Sigma 7

SOURCE LANGUAGE: Extended Fortran IV

PROGRAM CATEGORY: Utility

DESCRIPTION:

Some kinds of plankton tows include samples from different depths. It is desired to create an additional sample, combining data from each depth. For each species found in the tow, the program finds the number caught per 1000 cubic meters of water filtered, per 1000 meters of depth. The minimum, average, and maximum values of temperature, salinity, etc. are found, and the total biomass and volume of water filtered are found. The depth ranges of the integrated sample are the minimum and maximum depths of all the samples processed. The depth code of the integrated sample is '99'. An output file is created, containing the input file information, and the integrated samples.

INPUT:

The file containing input data is read through the F:1 DCB. This file contains four different kinds of records, distinguished by the first four bytes of the record. The kinds of records, and record identifiers, are:

T	Tow record
S1	Required (first) sample record
S2	Optional (second) sample record
C	Catch record

These records must be in the following order:

```
Tow record for first tow
     Sample record 1 for first sample of tow
     Sample record 2 for first sample of tow
          Catch record for sample
          Catch record for sample
            :
            :
     Sample record 1 for second sample of tow
     Sample record 2 for second sample of tow
          Catch record for sample
          Catch record for sample
            :
            :
```

```
Tow record for second tow
     Sample record 1 for first sample of tow
     Sample record 2 for first sample of tow
          Catch record for sample
          Catch record for sample
           :
           :
```

The formats of the different kinds of records are given in Tables 1, 2, 3, and 4.

OUTPUT:

The output file is created through the F:2 DCB. The format is identical to the format of the input file. The records for the integrated sample for each tow are added at the end of the other samples for the tow.

USAGE:

The job to load and run the program could be:

```
!JOB
!FORTRAN LS,GO
     source decks of PLANKINT and LJUST
!SET F:1 /infil
!SET F:2 /outfil;OUT;SAVE
!LYNX $;.3
!RUN
```

where 'infil' and 'outfil' should be replaced by the names of the input and output files respectively.

RESTRICTIONS:

1. The program does not check to be sure there are samples covering all depths between the minimum and maximum depths.

2. The average temperature, salinity, etc. for the integrated sample are found by averaging the minimum and maximum values of the integrated sample.

3. The input file must not contain sample records with a depth code of '99'.

STORAGE REQUIREMENTS: The program requires 3456 locations.

SUBPROGRAMS REQUIRED: MOVE and COMPAR from the account 3 library. LJUST included with this program.

OPERATIONAL ENVIRONMENT:

Device	Function	Special requirements
Disk	input	F:1 DCB
Disk	output	F:2 DCB

ERRORS & DIAGNOSTICS:

There are two kinds of error messages. The first kind are for input-output errors, and the second kind for problems with the input file. All error messages result in the program being terminated.

Input/output error messages are:

1. ERROR nn IN FIRST RECORD

   There was a read error of some kind in the first record.

2. OUTPUT ERROR or BUFFEROUT ERROR

   A write error.

3. READ ERROR

   Read error other than first record.

Error messages indicating problems with the input file are:

4. FIRST RECORD NOT TOW RECORD nnnn

   The program prints the first 4 characters of the offending record.

5. NOT S1 RECORD nnnn

   The record after a tow record must always be a S1 record. If not, this message appears. Again, the first 4 characters are printed.

6. ILLEGAL RECORD TYPE AFTER S1 nnnn

   The only legal record types after S1 are S2 or C.

7. ILLEGAL RECORD TYPE AFTER C nnnn

   Must be record type T, S1, or C.

8. DEPTH CODE 99 IN INPUT FILE

The input file must not contain a sample with a depth code '99'.

<u>PROGRAMMER</u>:	Mary Hunt
<u>ORIGINATOR</u>:	Peter Wiebe
<u>DATE</u>:	June, 1979
<u>REFERENCES</u>:	Documentation for Wiebe Database, M. Hunt.

VII. Program to Update Database

NAME: ZOOPUP

TYPE: Main program

PURPOSE: To update the ZOOSTOR database.

MACHINE: Sigma 7

SOURCE LANGUAGE: Extended Fortran IV

PROGRAM CATEGORY: Utility

DESCRIPTION:

Program ZOOPUP is written to update the ZOOSTOR database. It is assumed that the user has read the database description and other related documentation. The program includes a different procedure for each anticipated updating task. These tasks are:

1. Add Family, Genus, and Species names
2. Add Tow, Sample, and Catch information
3. Modify or delete Tow occurrences
4. Modify or delete Sample occurrences
5. Modify or delete Catch occurrences

The program includes two read subroutines, one to read Family/Genus/Species information, and one to read Tow/Sample/Catch information. The expected input for these routines is described under INPUT. Data in a different format can be used by supplying a new version of the corresponding routine. Nearly all communication between the read routines, the main program, and the Database Management routines, is through COMMON. A complete description of COMMON is included in this report. A description of the individual procedures follows.

Contents of COMMON

NAME	NO. WORDS	TYPE	CONTENTS
ICCB	14	INTEGER	SHOULD NOT BE CHANGED BY USER
ISETABL	36	INTEGER	NOT NEEDED BY USER PROGRAMS
IARTABL	2	INTEGER	NOT NEEDED BY USER PROGRAMS
ZOOHEAD	2	INTEGER	USER DOES NOT NEED THIS EITHER
TOWDATA			
CRUISE	1	INTEGER	CRUISE NAME, 4 CHARACTERS ALPHA
TOWNUM	1	INTEGER	TOW NUMBER, 4 CHARACTERS ALPHA
TOWTYP	1	INTEGER	TOW TYPE, 4 CHARACTERS ALPHA
FAMCODE	1	INTEGER	FAMILY CODE, ONE BIT PER FAMILY IDENTIFIED
YEAR	1	INTEGER	LAST 2 DIGITS OF YEAR OF TOW
MONTH	1	INTEGER	MONTH NUMBER OF TOW
DAY	1	INTEGER	DAY OF MONTH
TIME	1	INTEGER	TIME OF DAY, 24-HOUR CLOCK FORMAT
LONGTUDE	1	REAL	LONGITUDE IN DEGREES, + FOR EAST, - FOR WEST
LATUDE	1	REAL	LATITUDE IN DEGREES, + FOR NORTH, - FOR SOUTH
REGION	1	INTEGER	REGION OF TWO, 4 CHARACTERS ALPHA
NITEDAY	1	INTEGER	DAY-NIGHT CODE, D FOR DAY, N FOR NIGHT
KURR200	2	INTEGER	POINTERS USED BY DBM
SAMPLE			
DEPCODE	1	INTEGER	DEPTH CODE, '99' MEANS INTEGRATED SAMPLE
DEPTHS	2	INTEGER	MINIMUM AND MAXIMUM DEPTH OF SAMPLE IN METERS
TEMPS	3	REAL	MIN., AVG., AND MAX. TEMPERATURE OF SAMPLE
SALTS	3	REAL	MIN., AVG., AND MAX. SALINITY OF SAMPLE
OXYGEN	3	REAL	MIN., AVG., AND MAX. OXYGEN OF SAMPLE
LIGHT	3	REAL	MIN., AVG., AND MAX. LIGHT OF SAMPLE
CHLRPHYL	3	REAL	MIN., AVG., AND MAX. CHLOROPHYL OF SAMPLE
BIOMASS	1	REAL	TOTAL BIOMASS OF SAMPLE
ALIQUOT	1	REAL	FRACTION OF SAMPLE STUDIED, 8 MEANS 1/8
VOLFIL	1	REAL	VOLUME OF WATER FILTERED, IN CUBIC METERS
UNDEF	5	INTEGER	ROOM FOR EXPANSION
KURR210	2	INTEGER	POINTERS USED BY DBM
CATCH			
NPCUM	1	REAL	NUMBER PER CUBIC METER
KURR400	1	INTEGER	POINTER USED BY DBM
FAMILY			
FAMNAME	5	INTEGER	17-CHARACTER FAMILY NAME
KURR300	1	INTEGER	DBM POINTER
GENUS			
GENAME	5	INTEGER	18-CHARACTER GENUS NAME
KURR310	1	INTEGER	DBM POINTER
SPECIES			
SPENAME	4	INTEGER	16-CHARACTER SPECIES NAME
KURR320	2	INTEGER	POINTERS USED BY DBM
IARMAST	6	INTEGER	NOT NEEDED BY USER PROGRAMS

The first procedure is used to add Family, Genus, and Species names to the database. This procedure takes the following steps:

1. Initialize counters and pointers.
2. Call read subroutine and check status.
3. If Family name is different from previous Family, and is not already in the database, add it to the database.
4. If Genus name is different from previous Genus, and is not already in the database, add it to the database.
5. If Species is not already in the database, add Species name to the database.
6. Return to step 2.

Operation will be most efficient if the species to be added are already sorted by Family name and Genus name.

The next procedure is used to add tow data, sample data, and catch data. This procedure takes the following steps:

1. Initialize pointers and counters.
2. Call read subroutine and check status.
3. If cruise or tow number is different from that of the previous tow, and there is no tow occurrence with that cruise and tow number, add the tow occurrence to the database.
4. If the depth code of the current sample is different from the depth code of the previous sample, or the tow is not the same as the previous tow, the program checks to see if the specified tow already has a sample with the specified depth code. If not, the sample occurrence is added to the database. If such a sample already exists, a message is printed, and processing continues.
5. If the first 4 characters of the Genus name are blank, returns to step 2.
6. Checks to see if the sample already has a catch occurrence of the specified Genus-Species. If such an occurrence already exists, a message is printed, and the program returns to step 2.
7. If the specified Genus/Species is not in the database, prints a message, and returns to step 2.
8. Stores catch occurrence, linked to sample occurrence and Genus-Species occurrence.
9. Returns to step 2.

The primary purpose of this procedure is to add Tow, Sample, and Catch occurrences in the same job. The same results can be obtained by adding the Tow and Sample occurrences in one job, and adding the Catch occurrences later. If this method is used, a message will be printed for each sample, indicating that the occurrence already exists in the database, but the program will continue processing. It should be noted that the Family, Genus, and Species occurrences must already exist in the database.

The last three procedures are used to modify or delete Tow, Sample, or Catch occurrences. It is to be hoped that the use of these procedures will be kept to a minimum by careful checking of input before adding information to the database. The first of these procedures is used to modify (or delete) tow occurrences. Since a tow is identified by cruise and tow number, these two fields cannot be changed. This procedure takes the following steps:

1. Initialize pointers and counters.
2. Call read subroutine and check status.
3. Access tow occurrence with specified cruise and tow number.
4. Modify occurrence by replacing all fields with contents of corresponding COMMON locations. (Or delete occurrence.)
5. Return to step 2.

The next procedure is used to modify (or delete) samples. Since samples are identified by depth code, this field cannot be changed. In addition to sample information, the read subroutine must specify cruise and tow number, to identify the tow to which the sample belongs. The procedure takes the following steps:

1. Initialize pointers and counters.
2. Call read routine and check status.
3. If cruise and tow number are different from those of the previous tow, access the specified tow occurrence.
4. Find the sample occurrence with the specified depth code.
5. Modify the sample occurrence by replacing all fields with the contents of the corresponding COMMON locations. (Or delete occurrence.)
6. Return to step 2.

The last procedure is used to modify (or delete) catch occurrences. To properly identify the occurrence to be modified, the read subroutine must input the cruise and tow number of the tow, depth code of the sample, and Genus and Species names, in addition to the correct catch information. The steps taken are:

1. Initialize counters and pointers.

2. Call read subroutine and check status.

3. If cruise and tow number are different from those of the previous tow, access the specified tow occurrence.

4. Find sample occurrence with the specified depth code.

5. Access the catch occurrences from the sample one by one until we reach the catch occurrence of the specified Genus and Species.

6. Modify the catch occurrence by replacing all fields with the contents of the corresponding COMMON locations. (Or delete occurrence.)

7. Return to step 2.

There is no substitute for careful checking and rechecking before adding information to the database. If a tow occurrence is deleted, all sample occurrences for the tow are deleted also. If a sample occurrence is deleted, all catch occurrences for the sample are deleted. When a catch occurrence is deleted, the linkage for each species occurrence must be changed. All this is done by the Data Base Management routines, but shows that deletions are not recommended. Again, careful checking is necessary to avoid these time-consuming procedures.

INPUT:

A. Cards through F:105

One record is read from logical unit 105 for each procedure requested by the user. The record contains one or two keywords to identify the desired procedure, as follows:

cols.	codes	procedure
1-3	FAM	Add Family, Genus, and Species names
1-3	TOW	Add Tow, Sample, and Catch occurrences
1-7	MOD TOW MOD SAM MOD CAT	Modify already existing Tow, Sample, or Catch occurrences
1-7	DEL TOW DEL SAM DEL CAT	Delete existing Tow, Sample, or Catch occurrences
1-3	END	No more procedures

B. Family, Genus, and Species names through F:1

Input for the FAM procedure is done by Subroutine READSPEC. This routine expects one record for each species to be added to the database. These records should be presorted by Family and Genus. The format is:

cols.	Contents
1-17	Family name
21-38	Genus name
41-56	Species name

C. Tow, Sample, and Catch information through F:2

Input for all procedures except FAM is done by Subroutine READTOW. The amount of input needed depends on the procedure, but the format of the file is the same in any case.

This file is described in attached documentation, Section V.

D. Subschema file F:SSCH DCB

The user does not need to do anything about this, except include a SET command in his job.

E. Database F:DB01 DCB

The user must include a SET command in the job.

OUTPUT: A. At the conclusion of each procedure, the number of occurrences of each group added, modified, or deleted, is output on the printer. Error messages may also be printed. See ERRORS & DIAGNOSTICS, below.

B. The information supplied by the user is stored in the database.

USAGE: The job file below will take the following steps:

1. Compile ZOOPUP and associated subroutines from cards and create a load module named ZOOPUP.

2. It will add occurrences to the Family, Genus, and Species groups. Records containing the information are read through logical unit 1, which is assigned to file FAMFILE.

3. It will add occurrences to the Tow, Sample, and Catch groups. The required information is read through logical unit 2, which is assigned to file TOWCATFIL.

```
  !JOB
  !LIMIT (CORE,20),(TIME,3)
  !SET F:DB01 /ZOOSTOR
  !SET F:SSCH /ZOOPLAN
  !SET F:1 /FAMFILE;IN
  !SET F:2 /TOWCATFIL;IN
* !FORTRAN GO,NS
*      source of ZOOPUP and subroutines
* !LYNX $,DCB1.DMSLIB;.DMSLIB;.3 OVER ZOOPUP
  !RUN (LMN,ZOOPUP)
  !DATA
  FAM
  TOW
  END
```

* When ZOOPUP is loaded, delete these.

RESTRICTIONS:

1. Cruise name and tow number of tow occurrences cannot be changed. Depth code of sample occurrences cannot be changed.

2. Family, Genus, and Species must be added before Tow, Sample, and Catch occurrences.

STORAGE REQUIREMENTS:

The load module requires 16,896 locations, but additional core is required at run-time. The program will run in 20K.

SUBPROGRAMS REQUIRED:

A. The following have been written as part of the program:

FAMADD	To add Family, Genus, and Species occurrences
TOWADD	To add Tow, Sample, and Catch occurrences
FIXIT	To modify or delete already existing Tow, Sample, or Catch occurrences
READSPEC	To read the Family/Genus/Species file
READTOW	To read the Tow/Sample/Catch file
LJUST	To left-justify an alpha field within a word.

B. The following are from the Data Base Manager:

OPENUPD	STORE	SETERR	FINDG	FINDN	GET	CLOSEDB
HEAD	FINDD	MODIFY	DELETE			

C. The following are from the IPC library in account 3:

ABORT COMPAR MOVE

OPERATIONAL ENVIRONMENT:

Device	Function	Special requirements
card reader	control input	F:105 DCB
disk file	data input	F:1 DCB;IN
disk file	data input	F:2 DCB;IN
Subschema file	information input	F:SSCH DCB
Database file	data input/output	F:DB01 DCB
line printer	output	F:108 DCB

TIMING:

The following runs were done on a database of 100 pages, rather than 1000. This should probably not make much difference in the time.

1. Time to add 34 species records was .028 minutes.

2. Time to add 187 catch occurrences, from 3 tows and 27 samples, was .35 minutes.

ERRORS & DIAGNOSTICS:

A. The first two error messages are produced by the main program, and will result in termination of the program.

1. UNKNOWN PROCEDURE _____ _______

   The program did not recognize the procedure requested.

2. HEADER ERROR CODE = __________

   An error condition was returned by the DBM. The error code is printed.

B. The following messages are output by one of the read routines. The name of the subroutine is always printed. They will result in the termination of the current procedure.

3. SUBR. READSPEC ERROR IN READING FAMILY FILE

4. SUBR. REATOW ERROR IN READING TOW FILE

5. SUBR. READTOW FILE OUT OF ORDER
   HOWMUCH = _______
   LAST = __________
   CURRENT = _______

   The output values are for debugging purposes.

C. The final three messages indicate an unexpected condition in the database. The name of the subroutine is included as part of the message.

6. NON-EXISTENT GROUP OCCURRENCE

   This message is followed by identification of the group occurrence which cannot be located. In most cases, the current procedure will continue.

7. ERROR <u>error code</u> IN <u>subroutine name</u> <u>group name</u>

   This message indicates that an error condition was found by one of the DBM subroutines. The DBM error code, name of the subroutine, and the group in which the error occurred, are part of the error message. In most cases, the current procedure will be terminated.

8. GROUP OCCURRENCE ALREADY IN DATABASE

   This message is followed by identification of the group occurrence which is duplicated. The current procedure will continue.

<u>PROGRAMMER</u>:	Mary Hunt
<u>ORIGINATOR</u>:	Peter Wiebe
<u>DATE:</u>	October, 1979
<u>REFERENCES</u>:	Xerox Extended Data Management System Reference Manual.

VIII. Interactive Database Processor (IDP)

Most information retrieval can be done by the Interactive Database Processor, hereafter called IDP. This processor can be used either to print reports directly or to store retrieved information in a file for further processing. IDP is fairly flexible, easy to use, and well-documented. It requires 12K of core, and can be run either in batch or on-line. To use IDP, it is necessary to be familiar with the database structure and item-names, which are included in Section II of this report.

If IDP is used to store retrieved information in a file, it is recommended that the NON-REPORT option not be used. After the retrieval has been completed, use EDIT to eliminate column headings, and to determine the format of the records. The NON-REPORT option causes all items which are binary or floating-point to be stored in binary format, which might cause trouble with some application programs.

A word of caution about using IDP - be sure to give the search a place to start. If you don't, it will take much longer than it should. It may help to know that group ZOOHEAD is stored on page 1, line 1. If no other starting place is given or implied, include (in the DISPLAY command) the phrase DIRECT ON ZOOHEAD (0001,01). If you get the message 'UNABLE TO OPTIMIZE', hit BREAK and start over, giving more information.

```
!IDP
  IDP VERSION B02
:QUERY ZOOPLAN.980  AREA=ZOOSTOR.980.
:DISPLAY CRUISE, TOWNUM, TOWTYP, DEPCODE, BIOMASS
:WHEN CRUISE EQ 'K065' AND DEPCODE EQ '99'
:DIRECT ON ZOOHEAD (0001,01).

 CRUISE    TOWNUM    TOWTYP    DEPCODE       BIOMASS

  K065       76       MOC1        99       +8.6100000E+02
  K065       75       MOC1        99       +5.4400000E+02
  K065       73       MOC1        99       +1.2100000E+03
  K065       72       MOC1        99       +8.8700000E+02
  K065       71       MOC1        99       +9.9700000E+02
  K065       70       MOC1        99       +7.5500000E+02
  K065       69       MOC1        99       +9.5500000E+02
  K065       67       MOC1        99       +6.3700000E+02
  K065       66       MOC1        99       +5.3200000E+02
  K065       65       MOC1        99       +5.1800000E+02
  K065       64       MOC1        99       +5.4000000E+02
  K065       63       MOC1        99       +1.1330000E+03
  K065       62       MOC1        99       +1.2210000E+03
  K065       61       MOC1        99       +6.7300000E+02
  K065       60       MOC1        99       +4.9900000E+02

:SORT GENAME, SPENAME.
:DISPLAY FAMNAME, GENAME, SPENAME
:WHEN FAMNAME EQ 'EUPHAUSIDS'.
SORT VERSION  F03WHDI JUN 4 79
SEQUENTIAL
RECORDS IN TOURNAMENT:              76
NUMBER OF MERGE BUFFERS:            12
INTERMEDIATE BUFFER SIZE:          512
RECORDS INPUT:                      34
RECORDS OUTPUT:                     34

     FAMNAME               GENAME               SPENAME

 EUPHAUSIDS          BENTHEUPHAUSIA          AMBLYOPS
 EUPHAUSIDS          EUPHAUSIA               AMERICANA
 EUPHAUSIDS          EUPHAUSIA               BREVIS
 EUPHAUSIDS          EUPHAUSIA               GIBBOIDES
 EUPHAUSIDS          EUPHAUSIA               HEMIGIBBA
 EUPHAUSIDS          EUPHAUSIA               KROHNII
 EUPHAUSIDS          EUPHAUSIA               MUTICA
 EUPHAUSIDS          EUPHAUSIA               PSEUDOGIBBA
 EUPHAUSIDS          EUPHAUSIA               TENERA
 EUPHAUSIDS          MEGANYCTIPHANES         NORVEGICA
 EUPHAUSIDS          NEMATOBRANCHION         BOOPIS
 EUPHAUSIDS          NEMATOBRANCHION         FLEXIPIES
 EUPHAUSIDS          NEMATOBRANCHION         SEXSPINOSUS
 EUPHAUSIDS          NEMATOSCELIS            ATLANTICA
 EUPHAUSIDS          NEMATOSCELIS            MEGALOPS
 EUPHAUSIDS          NEMATOSCELIS            MICROPS
 EUPHAUSIDS          NEMATOSCELIS            TENELLA
 EUPHAUSIDS          STYLOCHEIRON            ABBREVIATUM
 EUPHAUSIDS          STYLOCHEIRON            AFFINE
```

```
:DISPLAY SPENAME, NPCUM, CRUISE, TOWNUM
:WHEN GENAME EQ 'NEMATOBRANCHION' AND DEPCODE EQ '99'.

     SPENAME              NPCUM          CRUISE   TOWNUM

BOOPIS              +1.8999999E-01      EN11      95
BOOPIS              +1.1000001E-01      EN11      91
BOOPIS              +1.8000000E-01      EN11      87
BOOPIS              +2.5000000E-01      EN11      86
BOOPIS              +1.9999998E-01      EN11      85
BOOPIS              +1.8000000E-01      EN11      84
BOOPIS              +2.0999997E-01      EN11      83
BOOPIS              +2.6999998E-01      EN11      82
BOOPIS              +1.9999998E-01      EN11      81
BOOPIS              +1.8999999E-01      SUW1      9
BOOPIS              +3.5000002E-01      SUW1      1
BOOPIS              +3.9999997E-01      K035      006
BOOPIS              +1.0900001E+00      A101      016
BOOPIS              +1.1700000E+00      A101      015
BOOPIS              +4.6300001E+00      A101      014
BOOPIS              +1.5600004E+00      A101      013
BOOPIS              +3.9999997E-01      A101      012
BOOPIS              +3.8999998E-01      A101      011
BOOPIS              +3.6000001E-01      A101      010
BOOPIS              +3.2999998E-01      A101      009
BOOPIS              +4.3999999E-01      O007      0001
BOOPIS              +1.6499996E+00      A085      004
BOOPIS              +7.7999997E-01      K062      58
BOOPIS              +2.2000002E-01      K062      57
BOOPIS              +2.4000000E-01      K065      71
BOOPIS              +1.9999998E-01      K065      66
BOOPIS              +5.1999998E-01      K065      64
BOOPIS              +1.0000000E+00      K038      6
BOOPIS              +2.0000000E+00      K038      1
BOOPIS              +1.0000000E+00      A271      20
BOOPIS              +1.0000000E+00      A271      11
BOOPIS              +1.0000000E+00      A271      3
BOOPIS              +3.8999998E-01      K062      48
BOOPIS              +2.5000000E-01      K062      47
BOOPIS              +1.6000002E-01      K053      42
BOOPIS              +3.6000001E-01      K053      37
BOOPIS              +2.9000002E-01      K053      33
BOOPIS              +5.9999998E-02      K053      31
BOOPIS              +7.9000002E-01      C125      005
BOOPIS              +2.6999998E-01      C125      011
BOOPIS              +2.6999998E-01      C125      008
BOOPIS              +2.9000002E-01      C125      013
FLEXIPIES           +1.8000000E-01      EN11      87
FLEXIPIES           +3.8999998E-01      K035      005
FLEXIPIES           +3.0000001E-01      A101      005
FLEXIPIES           +2.5200004E+00      C111      007
FLEXIPIES           +1.8199996E+00      C111      020
FLEXIPIES           +7.6000003E+00      C111      013
FLEXIPIES           +2.5299997E+00      C111      010
FLEXIPIES           +3.8999998E-01      K065      72
FLEXIPIES           +3.7999999E-01      K065      71
FLEXIPIES           +9.8000001E-01      K065      70
```

IX. Applications Programs

As an alternative to IDP, two subroutines have been written which can be incorporated in user applications programs. Their use will allow direct access to the database, but will add to the core requirements of the programs. These routines act as an interface between user programs and the Database Management routines. These will only access the data; they cannot be used to create or update the database. One of these routines accesses the database by Families, and the other accesses the database by Tows. Nearly all communication between the user programs, the access routines, and the Data Base Management routines is through COMMON. The contents and arrangement of COMMON are determined by the File Definition Processor, and must be exactly as specified. A complete description is included in this report. With the exceptions noted in the following descriptions (in GETFAM and GETTOW), the user program should change nothing in COMMON.

Each of the two routines has four entry points. In both cases, the first entry point is an initialization call. Although these routines are designed so that only one of them will be needed for most applications, both may be used in a single program, with the following restrictions:

1. Only one of the initialization calls may be made.
2. Calls to the two routines must not be alternated. Processing in one direction should be completed before starting to process in the other direction.

A brief demonstration program using FAMINIT is included. For each species, it prints tow and sample information associated with each catch occurrence. The first page of output is also included.

NAME: TOWINIT

TYPE: Subroutine

PURPOSE: To provide a method for Fortran programs to access the ZOOSTOR database by tow and sample number.

MACHINE: Sigma 7

SOURCE LANGUAGE: Extended Fortran IV

PROGRAM CATEGORY: Input

DESCRIPTION:

This routine was written to allow applications programs to access information stored in the ZOOSTOR database, without having to use the Data Base Manager directly. Nearly all communication between user programs, this access routine, and the Data Base Management routines is through COMMON. In particular, information retrieved from the database is always stored in COMMON. The contents and arrangement of COMMON are determined by the File Definition Processor, and must be exactly as specified. A complete description is included with this report. With the exceptions noted in the description (in GETTOW), the user program should change nothing in COMMON.

This routine has four entry points, the first of which is an initialization call. All the entry points include an error indicator, INKERR, as an argument. This indicator has the same meaning in all cases:

INKERR = 0	The requested operation was successfully completed.
INKERR = -1	The end of the set being processed has been found. This is not usually an error condition. It cannot occur in an initialization call.
INKERR > 0	An error has occurred which makes it impossible to continue. The value of INKERR will be one of the error codes set by the Data Base Manager. If the user program makes another call to the access routine after an error condition has been encountered, the job will be aborted.

INPUT: The input items requested by individual calls are stored in the corresponding locations of COMMON. See the description of COMMON included in this report.

OUTPUT: None.

USAGE:

A. As mentioned above, this routine has four entry points. The first entry point is an initialization call, which must be made once and only once, before any of the other entry points are called. It will open the database and do other needed initialization. The form of this call is:

```
CALL TOWINIT (NPAGE,INKERR)
```

where

NPAGE	is an integer location into which the calling program must store the number of pages to be used as buffers by the Data Base Manager. The value must be between 3 and 10.
INKERR	is an integer location into which the routine will store the results of the operation, as described above.

B. The next entry point is used to access a tow. It can be used in one of two ways: either to access a specific tow, or to access the next tow. To access the next tow, the calling program must set METHOD to one; the routine will store the retrieved tow information in the corresponding locations in COMMON. To access a specific tow, the calling program must store the cruise name and tow number in the corresponding locations in COMMON and set METHOD to zero before calling GETTOW. The form of the call is:

```
GETTOW (METHOD,INKERR)
```

where

METHOD	is an integer location into which the calling program must store either zero or one: = 0 to retrieve a specific tow = 1 to retrieve next tow
INKERR	is an integer location into which the routine will store the results of the operation, as described above.

C. The next call should be made to access the next Sample from the current Tow. When all samples for the tow have been retrieved, INKERR will be set to -1. Retrieved sample information will be stored in the corresponding locations of COMMON. The form of this call is:

```
CALL GETSAMP (INKERR)
```

where

INKERR	is an integer location into which the routine will store the results of the operation, as described above.

D. The last entry point of this routine is used to retrieve the next catch occurrence for the current sample. The calling program can specify how much, if any, information is desired in addition to the catch data. All retrieved data are stored by this routine in the corresponding locations in COMMON. The form of this call is:

```
CALL NECATCH (HOWMUCH,INKERR)
```

where

HOWMUCH	is an integer location into which the calling program must store a value to indicate how much information is desired: = 0 will access only catch information = 1 will also retrieve Genus and Species names = 2 will retrieve Family name in addition to Genus and Species names.
INKERR	is an integer location into which the routine will store the results of the operation, as described above.

E. Other Considerations

1. The job to access the database must include the following SET commands:

```
!SET F:DB01 /ZOOSTOR
!SET F:SSCH /ZOOPLAN
```

2. When the application program has completed its operation, it should close the database, as follows:

```
CALL CLOSEDB
```

3. The Load command for the application program must include the following:

   File DCB1 in account DMSLIB
   Library in account DMSLIB

   To load from the GO file, the LYNX command might be:

   ```
   !LYNX $,DCB1.DMSLIB;.DMSLIB;.3
   ```

RESTRICTIONS:

1. If the other retrieval routine, FAMINIT, is to be used in the same program, the following restrictions must be observed:

   a. Only one of the initialization calls may be made.

   b. Calls to the two routines must not be alternated. Processing in one direction should be completed before starting to process in the other direction.

2. The entry points are heirarchical. This means that a call to GETSAMP cannot be made before a call to GETTOW, and a call to NECATCH cannot be made before a call to GETSAMP.

3. The calling program should check the indicator after every call.

STORAGE REQUIREMENTS:

Subroutine TOWINIT requires 135 locations. This does not include locations needed by the Data Base Management routines.

SUBPROGRAMS REQUIRED:

The following routines are needed from the library in account DMSLIB:

FINDD FINDG FINDN GET HEAD OPENRET SETERR

OPERATIONAL ENVIRONMENT:

Device	Function	Special requirements
Subschema file ZOOPLAN	input	F:SSCH DCB
Database file ZOOSTOR	input	F:DB01 DCB

PROGRAMMER:	Mary Hunt
ORIGINATOR:	Peter Wiebe
DATE:	January, 1979
REFERENCES:	Xerox Extended Data Management System Reference Manual.

NAME: FAMINIT

TYPE: Subroutine

PURPOSE: To provide a method for Fortran programs to access the ZOOSTOR database by Family and Genus.

MACHINE: Sigma 7

SOURCE LANGUAGE: Extended Fortran IV

PROGRAM CATEGORY: Input

DESCRIPTION:

This routine was written to allow applications programs to access information stored in the ZOOSTOR database, without having to use the Data Base Manager directly. Nearly all communication between user programs, this access routine, and the Data Base Management routines is through COMMON. In particular, information retrieved from the database is always stored in COMMON. The contents and arrangement of COMMON are determined by the File Definition Processor, and must be exactly as specified. A complete description is included with this report. With the exceptions noted in the description (in GETFAM), the user program should change nothing in COMMON.

This routine has four entry points, the first of which is an initialization call. All the entry points include an error indicator, INKERR, as an argument. This indicator has the same meaning in all cases:

INKERR = 0	The requested operation was successfully completed.
INKERR = -1	The end of the set being processed has been found. This is not usually an error condition. It cannot occur in an initialization call.
INKERR > 0	An error has occurred which makes it impossible to continue. The value of INKERR will be one of the error codes set by the Data Base Manager. If the user program makes another call to the access routine after an error condition has been encountered, the job will be aborted.

INPUT: The input items requested by individual calls are stored in the corresponding locations of COMMON. See the description of COMMON included in this report.

OUTPUT: None.

USAGE:

A. As mentioned above, this routine has four entry points. The first entry point is an initialization call, which must be made once and only once, before any of the other entry points are called. It will open the database and do other needed initialization. The form of this call is:

```
CALL FAMINIT (NPAGE,INKERR)
```

where

NPAGE	is an integer location into which the calling program must store the number of pages to be used as buffers by the Data Base Manager. The value must be between 3 and 10.
INKERR	is an integer location into which the routine will store the results of the operation, as described above.

B. The next entry point is used to access a family. It can be used in one of two ways: either to access a specific family, or to access the next family. To access the next family, the calling program must set METHOD to one; the routine will store the name of the retrieved family in array FAMNAME in COMMON. To access a specific family, the calling program must store the Family name in array FAMNAME in COMMON and set METHOD to zero before calling GETFAM. The form of the call is:

```
CALL GETFAM (METHOD,INKERR)
```

where

METHOD	is an integer location into which the calling program must store either zero or one: = 0 to retrieve a specific family = 1 to retrieve next family
INKERR	is an integer location into which the routine will store the results of the operation, as described above.

C. The next call should be made to access the next Genus-Species in the current Family. The first time this call is made for a given Family, it will retrieve the first Genus in the Family, and the first Species for that Genus. Subsequent calls will retrieve the next Species for the same Genus until all Species in the Genus have been retrieved. It will then retrieve the first Species from the next Genus. When all Species from the Family have been retrieved, INKERR will be set to -1. Names of the retrieved Genus and Species will be stored in arrays GENAME and SPENAME in COMMON. The form of the call is:

```
CALL GETSPEC (INKERR)
```

where

INKERR	is an integer location into which the routine will store the results of the operation, as described above.

D. The last entry point of this routine is used to retrieve the next catch occurrence for the current Species. The calling program can specify how much, if any, information is desired in addition to the catch data. All retrieved data are stored by this routine in the corresponding locations in COMMON. The form of this call is:

```
CALL GETCATCH (HOWMUCH,INKERR)
```

where

HOWMUCH is an integer location into which the calling program must store a value to indicate how much information is desired.
= 0 will access only catch information
= 1 will also retrieve Sample which is the owner of the catch occurrence
= 2 will retrieve Tow information in addition to Sample information.

INKERR is an integer location into which the routine will store the results of the operation, as described above.

E. Other Considerations

1. The job to access the database must include the following SET commands:

```
!SET F:DB01 /ZOOSTOR
!SET F:SSCH /ZOOPLAN
```

2. [illegible]n the application program has completed its operation, [illegible]t should close the database: CALL CLOSEDB
This routine has no arguments.

3. The Load command for the application program must include the following:

   File DCB1 in account DMSLIB

   Library in account DMSLIB

   To load from the GO file, the LYNX command might be:

   ```
   !LYNX $,DCB1.DMSLIB;.DMSLIB;.3
   ```

RESTRICTIONS:

1. If the other retrieval routine, TOWINIT, is to be used in the same program, the following restrictions must be observed:
   a. Only one of the initialization calls may be made.
   b. Calls to the two routines must not be alternated. Processing in one direction should be completed before starting to process in the other direction.

2. The entry points are heirarchical. This means that a call to GETSPEC cannot be made before a call to GETFAM, and a call to GETCATCH cannot be made before a call to GETSPEC.

3. The calling program should check the indicator after every call.

STORAGE REQUIREMENTS:

Subroutine FAMINIT requires 146 locations. This does not include the locations needed by the Data Base Management routines.

SUBPROGRAMS REQUIRED:

The following routines are needed from the library in account DMSLIB:

FINDD FINDG FINDN GET HEAD OPENRET SETERR

OPERATIONAL ENVIRONMENT:

Device	Function	Special requirements
Subschema file ZOOPLAN	input	F:SSCH DCB
Database file ZOOSTOR	input	F:DB01 DCB

PROGRAMMER:	Mary Hunt
ORIGINATOR:	Peter Wiebe
DATE:	January, 1979
REFERENCES:	Xerox Extended Data Management System Reference Manual.

```
C     PROGRAM TO TEST FAMINIT
C
C
C*************************************************************
C
C       COMMON TO BE USED BY PROGRAMS WHICH ACCESS
C        ZOOSTOR DATABASE
C
        COMMON ICCB(14)
        INTEGER REFCODE, PAGENO, LINO, FREF
        INTEGER LAREF, GROUPNO, ERRCODE, ERRNO
        INTEGER ERREF, PASSWD(2), AREANO
        EQUIVALENCE (ICCB(1),REFCODE), (ICCB(2),PAGENO)
        EQUIVALENCE (ICCB(4),LINO), (ICCB(5),FREF)
        EQUIVALENCE (ICCB(6),LAREF), (ICCB(7),GROUPNO)
        EQUIVALENCE (ICCB(8),ERRCODE), (ICCB(9),ERRNO)
        EQUIVALENCE (ICCB(10),ERREF), (ICCB(11),PASSWD)
        EQUIVALENCE (ICCB(13),AREANO)
C
C     SET TABLES
C
        COMMON ISETABL(36)
        INTEGER TOWSET(5), SAMSET(5), FAMSET(5)
        INTEGER GENUSET(5), SPECSET(5), SPECAT(5)
        INTEGER SAMCAT(5)
        EQUIVALENCE (ISETABL(1),TOWSET), (ISETABL(6),SAMSET)
        EQUIVALENCE (ISETABL(11),FAMSET), (ISETABL(16),GENUSET)
        EQUIVALENCE (ISETABL(21),SPECSET), (ISETABL(26),SPECAT)
        EQUIVALENCE (ISETABL(31),SAMCAT)
C
        COMMON IARTABL(2)
C
C     HEADER GROUP
C
        COMMON ZOOHEAD(2)
        INTEGER KURR100
        EQUIVALENCE (ZOOHEAD(1),KURR100)
C
C     TOWDATA GROUP
C
        COMMON TOWDATA(14)
        INTEGER CRUISE, TOWNUM, TOWTYP
        INTEGER FAMCODE
        INTEGER YEAR, MONTH, DAY, TIME
        REAL LONGTUDE, LATUDE
        INTEGER REGION, NITEDAY, KURR200
        EQUIVALENCE (TOWDATA(1),CRUISE), (TOWDATA(2),TOWNUM)
        EQUIVALENCE (TOWDATA(3),TOWTYP), (TOWDATA(4),FAMCODE)
        EQUIVALENCE (TOWDATA(5),YEAR), (TOWDATA(6),MONTH)
        EQUIVALENCE (TOWDATA(7),DAY), (TOWDATA(8),TIME)
        EQUIVALENCE (TOWDATA(9),LONGTUDE), (TOWDATA(10),LATUDE)
        EQUIVALENCE (TOWDATA(11),REGION), (TOWDATA(12),NITEDAY)
        EQUIVALENCE (TOWDATA(13),KURR200)
C
C     SAMPLE GROUP
C
        COMMON SAMPLE(28)
        INTEGER DEPCODE, DEPTHS(2)
        REAL TEMPS(3), SALTS(3), OXYGEN(3)
```

```
      REAL LIGHT(3), CHLRPHYL(3), BIOMASS
      REAL ALIQUOT, VOLFIL
      INTEGER UNDEF(5), KURR210
      EQUIVALENCE (SAMPLE(1),DEPCODE), (SAMPLE(2),DEPTHS)
      EQUIVALENCE (SAMPLE(4),TEMPS), (SAMPLE(7),SALTS)
      EQUIVALENCE (SAMPLE(10),OXYGEN), (SAMPLE(13),LIGHT)
      EQUIVALENCE (SAMPLE(16),CHLRPHYL), (SAMPLE(19),BIOMASS)
      EQUIVALENCE (SAMPLE(20),ALIQUOT), (SAMPLE(21),VOLFIL)
      EQUIVALENCE (SAMPLE(22),UNDEF), (SAMPLE(27),KURR210)
C
C   CATCH GROUP
C
      COMMON CATCH(2)
      REAL NPCUM
      INTEGER KURR400
      EQUIVALENCE (CATCH(1),NPCUM), (CATCH(2),KURR400)
C
C   FAMILY GROUP
C
      COMMON FAMILY(6)
      INTEGER FAMNAME(5), KURR300
      EQUIVALENCE (FAMILY(1),FAMNAME), (FAMILY(6),KURR300)
C
C   GENUS GROUP
C
      COMMON GENUS(6)
      INTEGER GENAME(5), KURR310
      EQUIVALENCE (GENUS(1),GENAME), (GENUS(6),KURR310)
C
C   SPECIES GROUP
C
      COMMON SPECIES(6)
      INTEGER SPENAME(4), KURR320
      EQUIVALENCE (SPECIES(1),SPENAME), (SPECIES(5),KURR320)
C
C   AREA MASTER
C
      COMMON IARMAST(6)
C
C*************************************************************
C
C
C    LOCAL VARIABLES
C
      INTEGER NPAGE, INKERR, METHOD
      INTEGER HOWMUCH, NSTA
      REAL NUMTOT
      INTEGER IBLNK/'    '/
      INTEGER LP/108/
C
C                                   SET PARAMETERS AND MAKE
C                                    INITIALIZATION CALL
C
      NPAGE = 4
      NSTA = 0
      METHOD = 0
      HOWMUCH = 2
      CALL FAMINIT ( NPAGE, INKERR )
      IF ( INKERR .NE. 0 )  OUTPUT INKERR ; STOP 100
C
```

```
C                                  STØRE FAMILY NAME IN CØMMØN
C                                   AND RETRIEVE FAMILY
C
      FAMNAME(1) = 4HEUPH
      FAMNAME(2) = 4HAUSI
      FAMNAME(3) = 2HDS
      FAMNAME(4) = FAMNAME(5) = IBLNK
      CALL GETFAM ( METHØD, INKERR )
      IF ( INKERR .NE. 0 )  ØUTPUT INKERR ; STØP 200
C
C                                  SPECIES LØØP STARTS HERE
C
   20 CØNTINUE
      CALL GETSPEC ( INKERR )
      IF ( INKERR .LT. 0 )  GØ TØ 70
      IF ( INKERR .NE. 0 )  ØUTPUT INKERR ; STØP 300
      NUMTØT = 0
      WRITE(LP,2000)
 2000 FØRMAT(/)
      WRITE(LP,1000)  FAMNAME, GENAME, SPENAME
 1000 FØRMAT ( 2X,5A4,1X,5A4,1X,4A4 )
      NSTA = 0
C
C                                  LØØP THRØUGH CATCH ØCCURRENCES
C                                   FØR THIS SPECIES
C
   30 CØNTINUE
      CALL GETCATCH ( HØWMUCH, INKERR )
      IF ( INKERR .LT. 0 )  GØ TØ 20
      IF ( INKERR .NE. 0 )  ØUTPUT INKERR ; STØP 400
C
C                                  END ØF CURRENT SPECIES
C
   50 CØNTINUE
      WRITE(LP,1010)   CRUISE, TØWTYP, TØWNUM, YEAR, MØNTH, DAY
 1010 FØRMAT (1X,3A4, 3X,I2,'/',I2,'/',I2 )
      WRITE(LP,1020)  DEPCØDE, DEPTHS, TEMPS
 1020 FØRMAT(3X,A4, 2I8,   3F8.2 )
      GØ TØ 30
C
C                                  HAVE FINISHED THIS FAMILY, SØ...
C                                   STØP.
C
   70 CØNTINUE
      CALL CLØSEDB
      STØP
      END
```

Output from Demonstration Program

```
EUPHAUSIDS               THYSANØPØDA               MØNØCANTHA
K062MØC150          76/12/ 7
 99            1      1000     8•00     14•20     20•40
K062MØC150          76/12/ 7
 4           400       550    16•75     17•25     18•00

EUPHAUSIDS               THYSANØPØDA               ØBTUSIFRØNS
K062MØC145          76/12/ 4
 99            1      1000     5•00     12•70     20•30
K062MØC145          76/12/ 4
 5           300       400    13•25     14•00     14•75

EUPHAUSIDS               THYSANØPØDA               ØRIENTALIS

EUPHAUSIDS               THYSANØPØDA               PECTINATA

EUPHAUSIDS               THYSANØPØDA               TRICUSPIDATA

EUPHAUSIDS               THYSANØESSA               GREGARIA
K062MØC150          76/12/ 7
 99            1      1000     8•00     14•20     20•40
K062MØC150          76/12/ 7
 8             1       100    20•35     20•35     20•35
K062MØC147          76/12/ 5
 99            1      1000     5•00     12•70     20•40
K062MØC147          76/12/ 5
 8             1       100    20•35     20•35     20•35
K062MØC147          76/12/ 5
 3           550       700     7•50      8•50      9•50
K062MØC147          76/12/ 5
 2           700       850     5•75      6•25      7•50
K062MØC145          76/12/ 4
 99            1      1000     5•00     12•70     20•30
K062MØC145          76/12/ 4
 7           100       200    16•75     17•75     20•30
K062MØC145          76/12/ 4
 6           200       300    14•75     15•75     16•75
K062MØC145          76/12/ 4
 3           550       700     7•75      8•50     10•00

EUPHAUSIDS               THYSANØESSA               LØNGICAUDATA
K062MØC147          76/12/ 5
 99            1      1000     5•00     12•70     20•40
K062MØC147          76/12/ 5
```

X. Database Definition in DDL.

```
SCHEMA NAME IS ZOOSCHEM.
AREA NAME IS ZOOSTOR CONTAINS 1000 PAGES
              NUMBER IS 1.
     GROUP NAME IS ZOOHEAD
              WITHIN ZOOSTOR, RANGE IS PAGE 1 THRU PAGE 1
              LOCATION MODE IS DIRECT
              NUMBER IS 100.
     GROUP NAME IS TOWDATA
              WITHIN ZOOSTOR, RANGE IS PAGE 1 THRU PAGE 350
              LOCATION MODE IS CALC USING CRUISE, TOWNUM
              DUPLICATES ARE NOT ALLOWED
              NUMBER IS 200.
        CRUISE;     TYPE IS CHARACTER, 4.
        TOWNUM;     TYPE IS CHARACTER, 4.
        TOWTYP;     TYPE IS CHARACTER, 4.
        FAMCODE;    TYPE IS BINARY.
        YEAR;       TYPE IS BINARY.
        MONTH;      TYPE IS BINARY.
        DAY;        TYPE IS BINARY.
        TIME;       TYPE IS BINARY.
        LONGTUDE;   TYPE IS FLOATING SHORT.
        LATUDE;     TYPE IS FLOATING SHORT.
        REGION;     TYPE IS CHARACTER, 4.
        NITEDAY;    TYPE IS CHARACTER, 1.
     GROUP NAME IS SAMPLE
              WITHIN ZOOSTOR, RANGE IS PAGE 1 THRU PAGE 350
              LOCATION MODE IS VIA SAMSET SET
              NUMBER IS 210.
        DEPCODE;    TYPE IS CHARACTER, 4.
        DEPMIN;     TYPE IS BINARY.
        DEPMAX;     TYPE IS BINARY.
        TEMIN;      TYPE IS FLOATING SHORT.
        TEAVG;      TYPE IS FLOATING SHORT.
        TEMAX;      TYPE IS FLOATING SHORT.
        SALTMIN;    TYPE IS FLOATING SHORT.
        SALTAVG;    TYPE IS FLOATING SHORT.
        SALTMAX;    TYPE IS FLOATING SHORT.
        OXMIN;      TYPE IS FLOATING SHORT.
        OXAVG;      TYPE IS FLOATING SHORT.
        OXMAX;      TYPE IS FLOATING SHORT.
        LIGHTMIN;   TYPE IS FLOATING SHORT.
        LIGHTAVG;   TYPE IS FLOATING SHORT.
        LIGHTMAX;   TYPE IS FLOATING SHORT.
        CHLRMIN;    TYPE IS FLOATING SHORT.
        CHLRAVG;    TYPE IS FLOATING SHORT.
        CHLRMAX;    TYPE IS FLOATING SHORT.
        BIOMASS;    TYPE IS FLOATING SHORT.
        ALIQUOT;    TYPE IS FLOATING SHORT.
```

```
      VOLFIL;     TYPE IS FLOATING SHORT.
      UNDEF1;     TYPE IS BINARY.
      UNDEF2;     TYPE IS BINARY.
      UNDEF3;     TYPE IS BINARY.
      UNDEF4;     TYPE IS BINARY.
      UNDEF5;     TYPE IS BINARY.
    GROUP NAME IS CATCH
          WITHIN Z89STOR, RANGE IS PAGE 351 THRU PAGE 1000
          LOCATION MODE IS VIA SAMCAT SET, STORAGE IS SPECAT SET
          NUMBER IS 400.
      NPCUM;      TYPE IS FLOATING SHORT.
    GROUP NAME IS FAMILY
          WITHIN Z89STOR, RANGE IS PAGE 351 THRU PAGE 1000
          LOCATION MODE IS CALC USING FAMNAME
            DUPLICATES ARE NOT ALLOWED
          NUMBER IS 300.
      FAMNAME;    TYPE IS CHARACTER, 17.
    GROUP NAME IS GENUS
          WITHIN Z89STOR, RANGE IS PAGE 351 THRU PAGE 1000
          LOCATION MODE IS CALC USING GENAME
            DUPLICATES ARE NOT ALLOWED
          NUMBER IS 310.
      GENAME;       TYPE IS CHARACTER, 18.
    GROUP NAME IS SPECIES
          WITHIN Z89STOR, RANGE IS PAGE 351 THRU PAGE 1000
          LOCATION MODE IS VIA SPECSET
          NUMBER IS 320.
      SPENAME;    TYPE IS CHARACTER, 16.
    SET NAME IS TOWSET
      OWNER IS ZOOHEAD
          ORDER IS FIRST.
      MEMBER IS TOWDATA
          INCLUSION IS AUTOMATIC
          SELECTION IS THRU CURRENT OF SET.
    SET NAME IS SAMSET
      OWNER IS TOWDATA
          ORDER IS NEXT.
      MEMBER IS SAMPLE
          INCLUSION IS AUTOMATIC
          LINKED TO OWNER
          SELECTION IS THRU CURRENT OF SET.
    SET NAME IS FAMSET
      OWNER IS ZOOHEAD
          ORDER IS FIRST.
      MEMBER IS FAMILY
          INCLUSION IS AUTOMATIC
          SELECTION IS THRU CURRENT OF SET.
    SET NAME IS GENUSET
      OWNER IS FAMILY
          ORDER IS FIRST.
      MEMBER IS GENUS
          INCLUSION IS AUTOMATIC
          SELECTION IS THRU CURRENT OF SET.
    SET NAME IS SPECSET
      OWNER IS GENUS
          ORDER IS NEXT.
      MEMBER IS SPECIES
          INCLUSION IS AUTOMATIC
          LINKED TO OWNER
          SELECTION IS THRU CURRENT OF SET.
```

```
   SET NAME IS SPECAT
     OWNER IS SPECIES
         ORDER IS FIRST.
     MEMBER IS CATCH
         INCLUSION IS AUTOMATIC
         LINKED TO OWNER
         SELECTION IS THRU CURRENT OF SET.
   SET NAME IS SAMCAT
     OWNER IS SAMPLE
         ORDER IS NEXT.
     MEMBER IS CATCH
         INCLUSION IS AUTOMATIC
         LINKED TO OWNER
         SELECTION IS THRU CURRENT OF SET.
   END.
SUBSCHEMA NAME IS ZOOPLAN OF SCHEMA ZOOSCHEM
   COMPONENTS ARE ALL.
END.
```

Acknowledgements

Thanks to Woollcott Smith for providing advice in the preparation of the database and to the Information Processing Center at the Woods Hole Oceanographic Institution for providing the funds for M. Hunt to attend the EDMS applications programming class at Honeywell Education Center in Phoenix, Arizona. Thanks to Emily Evans for her care in typing this document. Funding for the preparation of this database system was provided by the Office of Naval Research Contract N00014-79-C-0071 NR083-004.

MANDATORY DISTRIBUTION LIST

FOR UNCLASSIFIED TECHNICAL REPORTS, REPRINTS, AND FINAL REPORTS
PUBLISHED BY OCEANOGRAPHIC CONTRACTORS
OF THE OCEAN SCIENCE AND TECHNOLOGY DIVISION
OF THE OFFICE OF NAVAL RESEARCH

(REVISED NOVEMBER 1978)

1 Deputy Under Secretary of Defense
(Research and Advanced Technology)
Military Assistant for Environmental Science
Room 3D129
Washington, D.C. 20301

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217
3 ATTN: Code 483
1 ATTN: Code 460
2 ATTN: 102B

1 CDR J. C. Harlett, (USN)
ONR Representative
Woods Hole Oceanographic Inst.
Woods Hole, MA 02543

Commanding Officer
Naval Research Laboratory
Washington, D.C. 20375
6 ATTN: Library, Code 2627

12 Defense Documentation Center
Cameron Station
Alexandria, VA 22314
ATTN: DCA

Commander
Naval Oceanographic Office
NSTL Station
Bay St. Louis, MS 39522
1 ATTN: Code 8100
1 ATTN: Code 6000
1 ATTN: Code 3300

1 NODC/NOAA
Code D781
Wisconsin Avenue, N.W.
Washington, D.C. 20235

Woods Hole Oceanographic Institution
WHOI-80-28

A DATABASE FOR ZOOPLANKTON NET TOW DATA by Mary M. Hunt and Peter H. Wiebe. June 1980. 65 pages. Prepared for the Office of Naval Research under Contract N00014-79-C-0071; NR 083-004.

This report describes the design and implementation of a database to store zooplankton net tow data and the applications programming done to update and access the database using the Sigma 7 Extended Database Management System. The database contains information about each tow (cruise name, tow number, type of tow, year, month, day, time of day, longitude, latitude, area of tow, day-night code); each sample (depth code, minimum and maximum depth; minimum, maximum and average values of temperature, salinity, oxygen, light and chlorophyll; total biomass; aliquot size; and volume of water filtered); each species (family, genus, species names, and catch per 1000 m^3). Information can be retrieved by user-written applications programs or with the Sigma 7 Interactive Database Processor, which can either print a report of the retrieved data or store it in a file for further processing. As presently formulated, the database can store up to 500 tows or samples, 10 families, 100 genera, 500 species and 50,000 catch records.

1. MOCNESS Net Tows
2. CODASYL Database for Zooplankton
3. Zooplankton Data Storage and Retrieval

I. Hunt, Mary M.
II. Wiebe, Peter H.
III. N00014-79-C-0071; NR 083-004

This card is UNCLASSIFIED

Woods Hole Oceanographic Institution
WHOI-80-28

A DATABASE FOR ZOOPLANKTON NET TOW DATA by Mary M. Hunt and Peter H. Wiebe. June 1980. 65 pages. Prepared for the Office of Naval Research under Contract N00014-79-C-0071; NR 083-004.

This report describes the design and implementation of a database to store zooplankton net tow data and the applications programming done to update and access the database using the Sigma 7 Extended Database Management System. The database contains information about each tow (cruise name, tow number, type of tow, year, month, day, time of day, longitude, latitude, area of tow, day-night code); each sample (depth code, minimum and maximum depth; minimum, maximum and average values of temperature, salinity, oxygen, light and chlorophyll; total biomass; aliquot size; and volume of water filtered); each species (family, genus, species names, and catch per 1000 m^3). Information can be retrieved by user-written applications programs or with the Sigma 7 Interactive Database Processor, which can either print a report of the retrieved data or store it in a file for further processing. As presently formulated, the database can store up to 500 tows or samples, 10 families, 100 genera, 500 species and 50,000 catch records.

1. MOCNESS Net Tows
2. CODASYL Database for Zooplankton
3. Zooplankton Data Storage and Retrieval

I. Hunt, Mary M.
II. Wiebe, Peter H.
III. N00014-79-C-0071; NR 083-004

This card is UNCLASSIFIED

Woods Hole Oceanographic Institution
WHOI-80-28

A DATABASE FOR ZOOPLANKTON NET TOW DATA by Mary M. Hunt and Peter H. Wiebe. June 1980. 65 pages. Prepared for the Office of Naval Research under Contract N00014-79-C-0071; NR 083-004.

This report describes the design and implementation of a database to store zooplankton net tow data and the applications programming done to update and access the database using the Sigma 7 Extended Database Management System. The database contains information about each tow (cruise name, tow number, type of tow, year, month, day, time of day, longitude, latitude, area of tow, day-night code); each sample (depth code, minimum and maximum depth; minimum, maximum and average values of temperature, salinity, oxygen, light and chlorophyll; total biomass; aliquot size; and volume of water filtered); each species (family, genus, species names, and catch per 1000 m^3). Information can be retrieved by user-written applications programs or with the Sigma 7 Interactive Database Processor, which can either print a report of the retrieved data or store it in a file for further processing. As presently formulated, the database can store up to 500 tows or samples, 10 families, 100 genera, 500 species and 50,000 catch records.

1. MOCNESS Net Tows
2. CODASYL Database for Zooplankton
3. Zooplankton Data Storage and Retrieval

I. Hunt, Mary M.
II. Wiebe, Peter H.
III. N00014-79-C-0071; NR 083-004

This card is UNCLASSIFIED

Woods Hole Oceanographic Institution
WHOI-80-28

A DATABASE FOR ZOOPLANKTON NET TOW DATA by Mary M. Hunt and Peter H. Wiebe. June 1980. 65 pages. Prepared for the Office of Naval Research under Contract N00014-79-C-0071; NR 083-004.

This report describes the design and implementation of a database to store zooplankton net tow data and the applications programming done to update and access the database using the Sigma 7 Extended Database Management System. The database contains information about each tow (cruise name, tow number, type of tow, year, month, day, time of day, longitude, latitude, area of tow, day-night code); each sample (depth code, minimum and maximum depth; minimum, maximum and average values of temperature, salinity, oxygen, light and chlorophyll; total biomass; aliquot size; and volume of water filtered); each species (family, genus, species names, and catch per 1000 m^3). Information can be retrieved by user-written applications programs or with the Sigma 7 Interactive Database Processor, which can either print a report of the retrieved data or store it in a file for further processing. As presently formulated, the database can store up to 500 tows or samples, 10 families, 100 genera, 500 species and 50,000 catch records.

1. MOCNESS Net Tows
2. CODASYL Database for Zooplankton
3. Zooplankton Data Storage and Retrieval

I. Hunt, Mary M.
II. Wiebe, Peter H.
III. N00014-79-C-0071; NR 083-004

This card is UNCLASSIFIED